PAT-NO: JP409231093A

DOCUMENT-IDENTIFIER: JP 09231093 A

TITLE: METHOD AND SYSTEM FOR TRANSFER OF PROGRAM CONTROL BETWEEN TWO ARCHITECTURES

PUBN-DATE: September 5, 1997

INVENTOR-INFORMATION:

NAME	COUNTRY
JOHN, W GOETTSU	
JOHN, M KETTY	
STEFAN, W MANN	

ASSIGNEE-INFORMATION:

NAME	COUNTRY
INTERNATL BUSINESS MACH CORP	N/A

APPL-NO: JP09024598

APPL-DATE: February 7, 1997

INT-CL (IPC): G06F009/46 , G06F009/455

ABSTRACT:

PROBLEM TO BE SOLVED: To provide a computer system which supports the transfer of control between two architectures in different methods for the call programs including the transfer of different word lengths and/or parameters, stack pointers and return addresses.

SOLUTION: The program control is transferred between two architectures in a control transfer mode bit. A bookkeeping system is generated in consideration of the call and return instructions between both architectures and holds the data received from the calling architecture to attain the different word length between the architectures. Then the stack pointers and return address of the master and slave call programs are stored in a register not in a main memory.

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号

特開平9－231093

(43)公開日　平成９年(1997)９月５日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ			技術表示箇所
Ｇ０６Ｆ	9/46	３６０		Ｇ０６Ｆ	9/46	３６０Ｂ	
	9/455				9/44	３１０Ａ	

審査請求　未請求　請求項の数38　ＯＬ　（全 14 頁）

(21)出願番号　特願平9－24598

(22)出願日　平成９年(1997)２月７日

(31)優先権主張番号　０８／６０５４０９

(32)優先日　1996年２月22日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　ジョン・ダブリュ・ゴエッツ
アメリカ合衆国05465、バーモント州ジェリコ、バターカップ・レーン　３

(74)代理人　弁理士　合田　潔　（外２名）

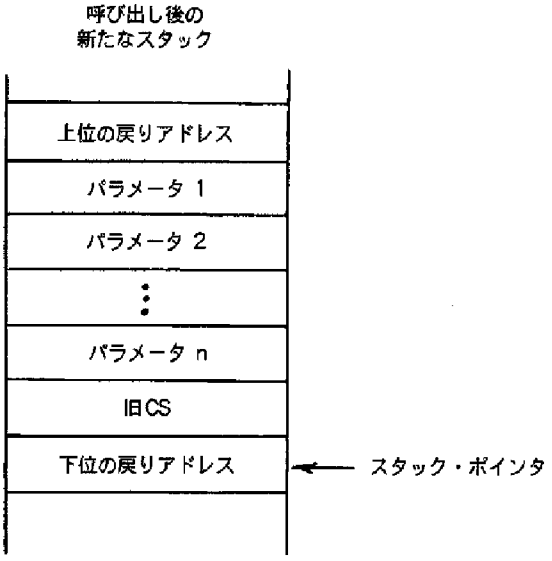
最終頁に続く

(54)【発明の名称】　２つのアーキテクチャ間でプログラム制御を転送する方法及びシステム

(57)【要約】

【課題】　異なるワード長及び（または）、パラメータ、スタック・ポインタ及び戻りアドレスの引き渡しを含む呼び出しプログラムにおいて、異なる方法で２つのアーキテクチャ間の制御転送をサポートするコンピュータ・システムを提供する。

【解決手段】　プログラム制御は制御転送モード・ビットによってアーキテクチャ間で転送される。ブックキーピング方式が２つのアーキテクチャ間の呼び出し及び戻りの命令を考慮して生成され、呼び出しアーキテクチャからのデータを保持し、アーキテクチャ間のワード長の違いを可能にする。マスタ／スレーブ関係の呼び出しプログラムのスタック・ポインタ及び戻りアドレスを、メイン・メモリではなくレジスタに格納する。



1

【特許請求の範囲】

【請求項１】ａ）第１のマイクロプロセッサ・アーキテクチャと、

ｂ）前記第１のマイクロプロセッサ・アーキテクチャとは異なる第２のマイクロプロセッサ・アーキテクチャと、

ｃ）前記第１及び第２のマイクロプロセッサ・アーキテクチャの、プログラム制御を解放する一方から獲得する他方に該プログラム制御を転送する手段と、

を含み、前記プログラム制御を転送する手段が、該プログラム制御を解放するマイクロプロセッサ・アーキテクチャから情報を受け取り、該プログラム制御を獲得するマイクロプロセッサ・アーキテクチャに前記情報を提供する、マイクロプロセッサ。

【請求項２】前記プログラム制御を転送する手段が、前記プログラム制御を獲得する前記マイクロプロセッサ・アーキテクチャが前記プログラム制御を解放する前記マイクロプロセッサ・アーキテクチャからのデータをメモリに必要としない場合、レジスタに該データを保持する手段を含む、請求項１記載のマイクロプロセッサ。

【請求項３】前記データを保持する手段が、前記プログラム制御を解放する前記マイクロプロセッサ・アーキテクチャの基底アドレスのオフセットを含む、請求項２記載のマイクロプロセッサ。

【請求項４】前記第２のマイクロプロセッサ・アーキテクチャが、前記第１のマイクロプロセッサ・アーキテクチャをエミュレートする手段を更に含む、請求項１記載のマイクロプロセッサ。

【請求項５】前記プログラム制御を転送する手段は、前記第１及び第２のマイクロプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す情報フィールドを含む、請求項１記載のマイクロプロセッサ。

【請求項６】外部メモリにある前記情報フィールドが、バスによって前記マイクロプロセッサに転送される、請求項５記載のマイクロプロセッサ。

【請求項７】前記情報フィールドが情報の１ビットを含み、前記ビットの状態が、前記第１及び第２のマイクロプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す、請求項５記載のマイクロプロセッサ。

【請求項８】前記第２のマイクロプロセッサ・アーキテクチャのアドレス範囲が、前記第１のマイクロプロセッサ・アーキテクチャのアドレス範囲より大きい、請求項１記載のマイクロプロセッサ。

【請求項９】前記第１のマイクロプロセッサ・アーキテクチャはＸ８６　ＣＩＳＣアーキテクチャである、請求項１記載のマイクロプロセッサ。

【請求項１０】前記第２のマイクロプロセッサ・アーキテクチャはPowerPC ＲＩＳＣアーキテクチャである、請求項１記載のマイクロプロセッサ。

2

【請求項１１】第１及び第２のプロセッサ・アーキテクチャ間でプログラム制御を転送する方法であって、

ａ）前記第１のプロセッサ・アーキテクチャに対応する第１のプログラムを与えるステップと、

ｂ）前記第２のプロセッサ・アーキテクチャに対応する第２のプログラムを与えるステップと、

ｃ）前記第１及び第２のプロセッサ・アーキテクチャを示す複数の状態を有する制御転送モード手段を与えるステップと、

ｄ）次に実行される命令を有する前記プログラムに対応するプロセッサ・アーキテクチャを示す前記複数の状態の１つに、前記制御転送モード手段を設定するステップと、

ｅ）制御転送データをメイン・メモリまたはハードウェア・レジスタのどちらに送信するかを決めるために、前記制御転送モード手段の状態を使用するステップと、

ｆ）前記制御転送モード手段の状態に従って、前記第１及び第２のプログラム間で制御転送データを送信するステップと、

を有する方法。

【請求項１２】前記制御転送モード手段は、情報の単一ビットの値にもとづいて前記第１及び第２のプロセッサ・アーキテクチャを示す、請求項１１記載の方法。

【請求項１３】前記第１のプロセッサ・アーキテクチャのアドレス範囲が前記第２のプロセッサ・アーキテクチャのアドレス範囲より小さい、請求項１１記載の方法。

【請求項１４】マイクロプロセッサ、及び外部バスで前記マイクロプロセッサに接続されている外部メモリを有するマイクロプロセッサ・システムの第１及び第２のプロセッサ・アーキテクチャ間にてプログラム制御を転送する方法であって、前記マイクロプロセッサは、前記外部メモリ、レジスタ及び実行ユニット間で接続されるバス及びキャッシュ・ユニットを有し、前記マイクロプロセッサは、自身が操作する前記レジスタにプロセス・スタックを格納するように構成され、前記方法は、

ａ）制御転送モード手段にて、前記第２のプロセッサ・アーキテクチャから前記第１のプロセッサ・アーキテクチャへの前記プログラム制御の転送を示すステップと、

ｂ）保持手段において前記第２のプロセッサ・アーキテクチャに関するデータを保持するステップと、

ｃ）前記プログラム制御を前記第２のプロセッサ・アーキテクチャから前記第１のプロセッサ・アーキテクチャに転送するステップと、

ｄ）前記第２のプロセッサ・アーキテクチャに関するデータを、前記プログラム制御の転送の際に前記保持手段から前記第２のプロセッサ・アーキテクチャに戻すステップと、

を含む方法。

【請求項１５】ｅ）前記制御転送モード手段を、前記外部バスを介して前記外部メモリに適用するステップと、

を含む、請求項１４記載の方法。
【請求項１６】前記プログラム制御を前記第２のプロセッサ・アーキテクチャから前記第１のプロセッサ・アーキテクチャに転送するステップが、
ｃ１）プログラムのアーキテクチャに従って、前記制御転送モード手段の状態を設定するステップと、
ｃ２）制御転送データが外部メモリまたはハードウェア・レジスタのどちらに送信されるかを決めるために前記制御転送モード手段の状態を使用するステップと、
ｃ３）前記制御転送手段の前記状態に従って、プログラム間で制御転送データを送信するステップと、
を含む、請求項１４記載の方法。
【請求項１７】前記制御転送モード手段が情報の１ビットで前記プログラム制御の転送を示す、請求項１４記載の方法。
【請求項１８】前記第１のプロセッサ・アーキテクチャのアドレス範囲が前記第２のプロセッサ・アーキテクチャのアドレス範囲より小さい、請求項１４記載の方法。
【請求項１９】前記プログラム制御転送手段が前記第１のプロセッサ・アーキテクチャから前記第２のプロセッサ・アーキテクチャへの前記プログラム制御転送を示す場合、前記保持手段に前記第１のプロセッサ・アーキテクチャの基底アドレスのオフセットを保持するステップと、
前記プログラム制御転送手段が前記第２のプロセッサ・アーキテクチャから前記第１のプロセッサ・アーキテクチャへの前記プログラム制御転送を示す場合、前記保持手段に前記第２のプロセッサ・アーキテクチャのアドレス・ビットを保持するステップと、
を更に含む、請求項１４記載の方法。
【請求項２０】マイクロプロセッサと外部メモリとを有するマイクロプロセッサ・システムであって、前記マイクロプロセッサは、前記外部メモリ、レジスタ及び実行ユニット間で接続されるバス及びキャッシュ・ユニットとを有し、前記マイクロプロセッサは、自身が操作する前記レジスタにプロセス・スタックを格納するように構成され、前記マイクロプロセッサ・システムは、
ａ）第１の連係方法に従って第１の連係データを格納するように構成される第１のマイクロプロセッサ・アーキテクチャと、
ｂ）第２の連係方法に従って第２の連係データを格納するように構成される第２のマイクロプロセッサ・アーキテクチャと、
ｃ）前記第１の連係データを前記第２の連係データに変換する手段と、
を有する、マイクロプロセッサ・システム。
【請求項２１】前記第２の連係データを前記第１の連係データに変換する手段を更に含む、請求項２０記載のマイクロプロセッサ・システム。
【請求項２２】前記連係データが第１及び第２のマイクロプロセッサ・アーキテクチャの一方によってメモリに記憶されることが必要ない場合、前記第１及び第２のマイクロプロセッサ・アーキテクチャの他方からの前記連係データをレジスタに保持する手段を更に含む、請求項２０記載のマイクロプロセッサ・システム。
【請求項２３】前記第２のマイクロプロセッサ・アーキテクチャは前記第１の連係方法をエミュレートする方法を更に含む、請求項２０記載のマイクロプロセッサ・システム。
【請求項２４】プログラム制御を転送する手段を含み、前記転送する手段が制御転送モード・フィールドを含み、前記フィールドが、前記第１及び第２のプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す、請求項２０記載のマイクロプロセッサ・システム。
【請求項２５】前記制御転送モード・フィールドは、前記マイクロプロセッサにより外部バスを介して前記外部メモリに書込まれる、請求項２４記載のマイクロプロセッサ・システム。
【請求項２６】前記制御転送モード・フィールドが情報の１ビットを含み、前記１ビットが、どのアーキテクチャが前記プログラム制御を有するかを示す、請求項２４記載のマイクロプロセッサ・システム。
【請求項２７】前記第２のマイクロプロセッサ・アーキテクチャのアドレス範囲が、前記第１のマイクロプロセッサ・アーキテクチャのアドレス範囲より大きい、請求項２０記載のマイクロプロセッサ・システム。
【請求項２８】前記第１のマイクロプロセッサ・アーキテクチャがＸ８６　ＣＩＳＣアーキテクチャである、請求項２０記載のマイクロプロセッサ・システム。
【請求項２９】前記第２のマイクロプロセッサ・アーキテクチャがPowerPC ＲＩＳＣアーキテクチャである、請求項２０記載のマイクロプロセッサ・システム。
【請求項３０】複数のマイクロプロセッサ及びメイン・メモリを有するマイクロプロセッサ・システムであって、前記マイクロプロセッサは、前記メイン・メモリ、レジスタ及び実行ユニット間で接続されるバス及びキャッシュ・ユニットを有し、自身が操作する前記レジスタにプロセス・スタックを格納するように構成され、前記マイクロプロセッサ・システムは、
ａ）第１のマイクロプロセッサ・アーキテクチャと、
ｂ）第２のマイクロプロセッサ・アーキテクチャと、
ｃ）前記第１及び第２のプロセッサ・アーキテクチャの、プログラム制御を解放する一方から獲得する他方に前記プログラム制御を転送する記憶手段と、
を含み、前記記憶手段が、前記プログラム制御を解放するプロセッサ・アーキテクチャからの情報を有し、該情報が、前記プログラム制御を獲得するプロセッサ・アーキテクチャによりアクセスされ得る、前記マイクロプロセッサ・システム。

5

【請求項３１】前記記憶手段は、前記プログラム制御を獲得する前記プロセッサ・アーキテクチャによって前記情報がメモリに記憶されることを必要としない場合、少くとも１つのレジスタに前記情報を記憶する、請求項３０記載のマイクロプロセッサ・システム。

【請求項３２】前記第２のアーキテクチャが前記第１のアーキテクチャをエミュレートする手段を更に有する、請求項３０記載のマイクロプロセッサ・システム。

【請求項３３】前記記憶手段は、前記第１及び第２のマイクロプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す制御転送モード・フィールドを有する、請求項３０記載のマイクロプロセッサ・システム。

【請求項３４】前記メイン・メモリにある前記制御転送モード・フィールドが、前記複数のマイクロプロセッサの１つに転送される、請求項３３記載のマイクロプロセッサ・システム。

【請求項３５】前記制御転送モード・フィールドが情報の１ビットを含み、前記１ビットが、どのアーキテクチャがプログラム制御を有するかを示す、請求項３３記載のマイクロプロセッサ・システム。

【請求項３６】前記第２のマイクロプロセッサ・アーキテクチャのアドレス範囲が、前記第１のマイクロプロセッサ・アーキテクチャのアドレス範囲より大きい、請求項３０記載のマイクロプロセッサ・システム。

【請求項３７】前記第１のマイクロプロセッサ・アーキテクチャがＸ８６　ＣＩＳＣアーキテクチャである、請求項３０記載のマイクロプロセッサ・システム。

【請求項３８】前記第２のマイクロプロセッサ・アーキテクチャがPowerPC ＲＩＳＣアーキテクチャである、請求項３０記載のマイクロプロセッサ・システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、一般にコンピュータ・システムの制御転送に関し、特に２つの異なるコンピュータ・アーキテクチャ間におけるコンピュータ・システムの制御転送に関する。

【０００２】

【従来の技術】最新のコンピュータ・アーキテクチャの多くは、一般に旧式のコンピュータ・アーキテクチャよりも大きな処理能力を実現している。たとえば、ＲＩＳＣプロセッサ・アーキテクチャを例にとると、Ｘ８６ＣＩＳＣアーキテクチャを含むＣＩＳＣプロセッサ・アーキテクチャに対して明確な性能優越性を有する実施形態を生み出した。しかしながら、Ｘ８６アーキテクチャなどの旧式のアーキテクチャのために開発された膨大な量のソフトウェアは、市場の新たなプロセッサの導入に対して大きな障害となり、このソフトウェア・ベースを新たなアーキテクチャへ移行させるには莫大な時間と資金を要し、危険性が伴う。多数のＲＩＳＣプロセッサ及びシステムのベンダーは、ソフトウェアのＸ８６アーキテクチャをエミュレートして、Ｘ８６遺産のソフトウェアを現行製品に移行させようとしている。

6

【０００３】ソフトウェア・エミュレーション技法においては重大な２つの欠点がある。１）ソフトウェア・エミュレーションはエミュレータ及びエミュレートされたプログラムを保持するのに必要なメモリにかなりの投資を必要とし、２）ソフトウェア・エミュレーションは本質的に速度が遅い。現在のソフトウェア・エミュレーション技術は、前の世代のＸ８６プロセッサのロー・エンド並の性能しか発揮せず、一般に１６メガバイトの物理的メモリを必要とする。

【０００４】ソフトウェア・エミュレーションにおける速度の問題に対する解決策の１つは、２つ以上のマイクロコントローラを並列処理で使用することである。米国特許番号第４３７０７０９号"Computer Emulator With Three Segment Microcode Memory And Two Separate Microcontrollers For Operand Derivation And Execution Phases"（１９８３年１月、Fosdickに付与、Tracor Inc.に譲渡）で開示される複数の内蔵型マイクロコントローラは、マクロ命令を２つのフェーズに分類することによって、より大きなマクロコントローラ命令をエミュレートする。１つのマイクロコントローラがマクロ命令を２つのフェーズに分類し、それから代行して２つの他のマイクロコントローラを制御して２つのフェーズのそれぞれを操作する。しかしながら、このシステムは、大型のコンピュータ・ハードウェアを必要とし、エミュレーション・プログラム及びエミュレータのために物理的なメモリにかなりの投資が必要とされるため、非常に高価である。

【０００５】複数のアーキテクチャ間のソフトウェア・プログラムの変換或いはエミュレーションをサポートする処理システムの他の例では米国特許番号第５２８０５８９号"Memory Access Control System For Use With As Relatively Small Size Data Processing System"（１９９４年１月発行、株式会社東芝に譲渡）、同第４５１４８０３号"Methods For Partitioning Mainframe Instruction Sets To Implement Microprocessor Based Emulation Thereof"（１９８５年４月発行、ＩＢＭに譲渡）、同第５２４７５２０号"Communications Architecture Interface"（１９９３年９月発行、ＩＢＭに譲渡）等がある。上述の全特許が本発明に関連しているので参照されたい。

【０００６】上述の開示されたコンピュータ・システムは、異なるアーキテクチャ間においてある種の通信が可能であるが、異なるワード長を有するアーキテクチャ間では通信はできない。PowerPCとＸ８６の場合、Ｘ８６アーキテクチャは現在３２ビット・アーキテクチャとして定義されており、一方PowerPCは３２ビット及び６４ビット・アーキテクチャの両方である。このことは、完

全な制御転送を容易にするために３２ビットＸ８６コードを６４ビットPowerPC空間に戻すことができなければならないことを意味する。更に、上述の特許群は、内部または外部メモリの実質的使用を必要とし、これによりプロセッサの処理速度が遅くなる。特にＸ８６の互換性は、パラメータ及びプロセス戻り情報の転送のために、一般にメモリでプロセス・スタックが必要となる。プロセス・スタックの使用は、データをメモリに格納したり、またメモリからロードする必要があるため、しばしば処理が遅くなり、性能を落とすことになる。

【０００７】

【発明が解決しようとする課題】従って本発明の目的は、異なるワード長及び連係方法（すなわちパラメータ、スタック・ポインタ及び戻りアドレスを送信する方法）を有するプロセッサ・アーキテクチャ間で、プログラム制御転送ができるコンピュータ・システムを提供することである。

【０００８】本発明の別の目的は、パラメータ及びプロセス戻り情報を転送するためにメモリ内でプロセス・スタックの使用を必要としないコンピュータ・システムを提供することである。

【０００９】本発明の別の目的は、制御転送に使用されるＸ８６アプリケーション或いは他の同様なアプリケーションに対して逆方向の互換性を提供することである。

【００１０】

【課題を解決するための手段】前述の及び本発明の他の目的は、複数のアーキテクチャ間においてプログラム制御を転送するマイクロプロセッサによって実現され、例えばＲＩＳＣとＸ８６ＣＩＳＣにおいて、記述子テーブル・エントリの１ビットを使用することにより、ターゲット戻りセグメントが第１のアーキテクチャまたは第２のアーキテクチャからのソフトウェアを含んでいるかどうかを示すことができる。第１のアーキテクチャの連係方法に使用されたが第２のアーキテクチャによって事前に指定されなかったレジスタを、第２のアーキテクチャによって使用して過剰のワード・ビットを保持することにより、両方のアーキテクチャが異なるワード長を使用できるようにする。更に、呼び出しプログラムのスタック・ポインタ及び戻りアドレスはメモリではなくレジスタに残され、そのため制御転送の性能が改良される。

【００１１】

【発明の実施の形態】図１を参照すると、本発明のコンピュータ・システムの概要が示されている。中央処理ユニット（ＣＰＵ）１はバス３を介してメイン・メモリ２と通信する。制御の転送中、ＣＰＵ１はメイン・メモリ２と要素の読出しまたは書込みを行い、或いは後で説明するレジスタ９と読出しまたは書込みを実行する。ＣＰＵ１は、バス及びキャッシュ・ユニット４、実行ユニット１１、命令エミュレーション・ユニット５、及び命令ディスパッチ及び完了ユニット８を有する。命令エミュレーション・ユニット５は適切な位置で命令を取り出すための命令フェッチ・ブロック６、及び命令がデコードされる命令デコード・ブロック７を有する。レジスタ９と命令ディスパッチ・ブロック１０は命令ディスパッチ及び完了ユニット８を形成する。

【００１２】また、バス３はＣＰＵ１を第２のＣＰＵに接続する。第２のＣＰＵアーキテクチャは第１のＣＰＵアーキテクチャとは異なるワード長、及び（または）パラメータ、スタック・ポインタ及び戻りアドレスの送信を含む異なる方法の呼び出しプログラムを有する。プロセッサ間にはマスタ／スレーブ関係が存在する。

【００１３】図５乃至図７はＸ８６アーキテクチャがどのようにインターレベル制御転送中、及びその前後においてプロセス・スタックの構造を指定するかを示す。特定の参照がＩＢＭのPowerPC ＲＩＳＣ及びＸ８６　ＣＩＳＣプロセッサ・アーキテクチャに対して行われるが、当業者は２つのアーキテクチャ・プロセッサを設計する場合、本発明がこれらの特定の詳細及び他のプロセッサ・アーキテクチャなしで実行できることは当業者には容易に理解できよう。

【００１４】制御転送が行われる前に、図５で示されるように呼び出しプログラムはパラメータ（最大３２バイトまで）をそのプロセス・スタックに置くことになる。このスタックの位置はスタック・ポインタＳＳ．ＥＳＰによって指定される。ここでＳＳはスタック・セレクタ・レジスタであり、ＥＳＰはスタックの基底アドレスからスタックまでのオフセットを保持するレジスタである。スタックの基底アドレスはセレクタＳＳによって参照されるスタックの記述子に含まれる。

【００１５】図６で示されるように、Ｘ８６プログラムが他のＸ８６プログラムを呼び出す場合、幾つかの項目が呼び出されたプログラムのスタックにプッシュされる。最初に呼び出しプログラムのスタック・ポインタ（ＳＳ．ＥＳＰ）が新たなスタックにプッシュされ、古いスタックからのパラメータが引き続く。最後に、呼び出しプログラムの戻りアドレス（ＣＳ．ＥＩＰ）がスタックにプッシュされる。ＣＳはコード・セレクタ・レジスタであり、ＥＩＰはコード・セグメント基底アドレスからのオフセットをコード・セグメントに保持するレジスタである。基底アドレスはコード記述子に含まれ、セレクタＣＳによって参照される。ＳＳ．ＥＳＰレジスタは新たなスタックの位置によって更新され、及びＣＳ．ＥＩＰレジスタは呼び出されたプログラムのアドレスによって更新されることに注意されたい。

【００１６】呼び出されたプログラムが呼び出し側に戻される場合、古いＣＳ．ＥＩＰの値は新たなスタックからポップオフされ、戻りアドレスによってＣＳ．ＥＩＰレジスタを更新するのに使用される。古いＳＳ．ＥＳＰ値はまた、新たなスタックからポップオフされ、最初に呼び出されたときのパラメータとして送信されたバイト

数だけ減らされて、ＳＳ．ＥＳＰレジスタを更新するのに使用される。図７は、戻り後の古いスタックの最終状態を示す。

【００１７】この形態のパラメータ、スタック・ポインタ、及び戻りアドレスを送信するプロセスは、まとめてＸ８６連係方法またはＸ８６連係規則と呼ぶことができる。

【００１８】PowerPCアーキテクチャは、プログラム制御転送において連係規則を明白に指定しないので、あるオペレーティング・システムは、仕事を遂行するのに必要な最適なものとして任意の規則を指定する可能性がある。しかしながら、人工遺物として、PowerPCからＸ８６への制御転送には、呼び出しPowerPCプログラムがＸ８６連係規則に従う必要がある。従ってそのような呼び出しは、古いスタックの存在に関係無く、直接呼び出されたＸ８６プログラムにプロセス・スタックを構築するためにPowerPCプログラムを必要とする。技術的に、古いスタック・ポインタ（ＳＳ．ＥＳＰ）は新たなプロセス・スタック内に置く必要はない。これはPowerPCアーキテクチャはＳＳまたはＥＳＰレジスタを指定しないからである。

【００１９】古いＳＳ．ＥＳＰ値は、PowerPC呼び出しの場合スタックに置く必要はないので、本発明ではその位置の目的を再指定し、３２ビットのＸ８６ソフトウェアから６４ビットのアドレス空間に戻る問題を解決する。再び図１を参照すると、呼び出しがPowerPCコードからＸ８６コードに行われる場合、PowerPCコードは６４ビットの戻りアドレスの上位３２ビットを、Ｘ８６とＸ８６間の制御転送の古いＥＳＰ用の予約位置に置く。戻りアドレスの下位３２ビットは、古いＥＩＰ位置に置かれる。これらのアドレス要素は、ＣＰＵとメモリ間のバス３を介してスタックに置かれる。Ｘ８６プログラムがPowerPCプログラムに戻る場合、プロセッサ・ハードウェアは、Ｘ８６プロセス・スタックの位置から再びバス３を介して戻りアドレスの上位及び下位をポップすることになる。バス・ユニット４は２つの要素を命令エミュレーション・ユニット５に転送し、ここでその２つの要素を連結して単一の６４ビット・アドレスを形成する。この６４ビット・アドレスは命令フェッチ・ブロック６によって使用されて新たな位置で命令を取り出し始め、効果的に制御をPowerPCプログラムに転送して戻す。

【００２０】図８はPowerPCプログラムがＸ８６プログラムを呼び出した後の再指定されたプロセス・スタック（呼び出し後の"新たな"スタック）を示す。３２ビットのPowerPCコードは戻りアドレスの上位３２ビットに単純に０を書込むので、Ｘ８６とPowerPCの３２ビットのアドレス空間は互いにオーバレイする。

【００２１】上述したように本発明において、プロセッサ・ハードウェアは、２つのメカニズムのうちのどちらを使用するかを、Ｘ８６プログラムから呼び出しプログラムに戻る際に決めねばならない。実質的には、戻りプログラムがPowerPCかＸ８６かであるかはソフトウェアによって告げられる。これはハードウェアがプログラムの戻りを実行する際に古いＣＳセレクタ及び関連する記述子テーブル・エントリを調べることによって行われる。このようにＸ８６アーキテクチャは、ＣＳセレクタがプロセス・スタックから読出される際、古いＣＳセレクタ値がプロセッサによってロードされたメイン・メモリの記述子を参照して知ることができる。記述子テーブル・エントリは、ＣＳセレクタによって参照されたメモリのセグメントについての情報を有する。この記述子エントリの１ビット、特に記述子の上位４バイト（ダブルワード）のビット位置２１は、３２ビットＸ８６アーキテクチャによって指定されないまま残る。本発明では、上位のダブルワードのビット位置２１を、ターゲット戻りセグメントが標準のＸ８６ソフトウェア（この時のビットは０）か、またはPowerPCソフトウェア（この時のビットは１）であるかどうかを示すものと定義する。ビット位置２１は以降、制御転送モード・ビットとして示される。図９、図１０及び図１１は、Ｘ８６アーキテクチャの３つの基本セグメント記述子、すなわち、データ・セグメント記述子、コード・セグメント記述子、及びシステムセグメント記述子のレイアウトを示す図である。また制御転送モード・ビット位置２１も示されている。

【００２２】従って、Ｘ８６の戻りにより、メイン・メモリにある関連する記述子を読取るのと同様に、プロセッサ・ハードウェアがメイン・メモリのプロセス・スタックから古いＣＳセレクタ値を読取る。命令エミュレーション・ユニット５（図１参照）は、発生した制御転送の形態を確かめるために、新たに指定された制御転送モード・ビットを調べる。制御転送モード・ビットが０であれば標準Ｘ８６制御戻りが生じている。制御転送モード・ビットが１であれば、戻りPowerPCアドレスの上位及び下位の３２ビットはプロセス・スタックから読取られ、フル６４ビットの戻りアドレスを形成する。それから制御はこのアドレスにある命令に転送される。

【００２３】この本発明の第２の態様は、Ｘ８６プログラムからPowerPCプログラムへの制御転送に関する。PowerPCアーキテクチャは固定された連係規則を指定しないので、Ｘ８６アーキテクチャの連係規則にも制限されない。Ｘ８６プロセス・スタックはPowerPCプログラムに送信されるパラメータを有するが、パラメータ及び古いＳＳ．ＥＳＰ及びＣＳ．ＥＩＰ値を有するPowerPCプログラムに対して新たなスタックを構築する必要はない。実際にこれらの値を、メモリに書込むのではなくプロセッサ・レジスタに直接残すことにより、Ｘ８６からPowerPCへの呼び出し性能は改良することができる。PowerPCからＸ８６への戻りは、戻りアドレスを最初にメモリか

ら読取るよりも、レジスタに保持しレジスタの戻りアドレスに直接分岐することによって改良できる。

【００２４】従って、本発明の第２の態様は、呼び出しＸ８６プログラムのスタック・ポインタ及び戻りアドレスをメモリではなくレジスタに置くことである。これは呼び出されたプログラムのアーキテクチャ、この場合PowerPCであるが、Ｘ８６アーキテクチャの連係制限に拘束されないからである。プロセッサはＸ８６とＸ８６間の制御転送と、Ｘ８６とPowerPC間の制御転送とを識別できなければならない。この識別は、転送（Ｘ８６のジャンプまたは呼び出し）を開始するときに使用される命令により参照されるゲート記述子をプロセッサが調べることによって達成される。このようにＸ８６アーキテクチャは、ゲート記述子が、ターゲット・コード・セグメントを指すセレクタを保持することがわかる。命令エミュレーション・ユニットは、ゲート記述子のセレクタによって参照されたコード・セグメント記述子を読取るが、これらの両方はメイン・メモリにある。コード・セグメント記述子は命令エミュレーション・ユニットによって読取られた後、制御転送モード・ビットを参照して次に発生する転送の種類を決める。制御転送モード・ビットが０の場合、通常の制御転送が発生する。制御転送モード・ビットが１の場合、PowerPCコードへの転送が生じ、戻りアドレス及び古いスタック・ポインタはメモリではなくプロセッサ・レジスタに残る。この実施形態でこれらの値を保持するために指定されるレジスタが図１２に示されている。

【００２５】本発明は、特に好適な実施例に対して図示、説明されたが、当業者は本発明の趣旨、及び範囲内で形状及び詳細を様々に変更できることが理解できよう。

【００２６】まとめとして、本発明の構成に関して以下の事項を開示する。

【００２７】（１）ａ）第１のマイクロプロセッサ・アーキテクチャと、

ｂ）前記第１のマイクロプロセッサ・アーキテクチャとは異なる第２のマイクロプロセッサ・アーキテクチャと、

ｃ）前記第１及び第２のマイクロプロセッサ・アーキテクチャの、プログラム制御を解放する一方から獲得する他方に該プログラム制御を転送する手段と、を含み、前記プログラム制御を転送する手段が、該プログラム制御を解放するマイクロプロセッサ・アーキテクチャから情報を受け取り、該プログラム制御を獲得するマイクロプロセッサ・アーキテクチャに前記情報を提供する、マイクロプロセッサ。

（２）前記プログラム制御を転送する手段が、前記プログラム制御を獲得する前記マイクロプロセッサ・アーキテクチャが前記プログラム制御を解放する前記マイクロプロセッサ・アーキテクチャからのデータをメモリに必要としない場合、レジスタに該データを保持する手段を含む、前記（１）記載のマイクロプロセッサ。

（３）前記データを保持する手段が、前記プログラム制御を解放する前記マイクロプロセッサ・アーキテクチャの基底アドレスのオフセットを含む、前記（２）記載のマイクロプロセッサ。

（４）前記第２のマイクロプロセッサ・アーキテクチャが、前記第１のマイクロプロセッサ・アーキテクチャをエミュレートする手段を更に含む、前記（１）記載のマイクロプロセッサ。

（５）前記プログラム制御を転送する手段は、前記第１及び第２のマイクロプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す情報フィールドを含む、前記（１）記載のマイクロプロセッサ。

（６）外部メモリにある前記情報フィールドが、バスによって前記マイクロプロセッサに転送される、前記

（５）記載のマイクロプロセッサ。

（７）前記情報フィールドが情報の１ビットを含み、前記ビットの状態が、前記第１及び第２のマイクロプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す、前記（５）記載のマイクロプロセッサ。

（８）前記第２のマイクロプロセッサ・アーキテクチャのアドレス範囲が、前記第１のマイクロプロセッサ・アーキテクチャのアドレス範囲より大きい、前記（１）記載のマイクロプロセッサ。

（９）前記第１のマイクロプロセッサ・アーキテクチャはＸ８６　ＣＩＳＣアーキテクチャである、前記（１）記載のマイクロプロセッサ。

（１０）前記第２のマイクロプロセッサ・アーキテクチャはPowerPC ＲＩＳＣアーキテクチャである、前記

（１）記載のマイクロプロセッサ。

（１１）第１及び第２のプロセッサ・アーキテクチャ間でプログラム制御を転送する方法であって、

ａ）前記第１のプロセッサ・アーキテクチャに対応する第１のプログラムを与えるステップと、

ｂ）前記第２のプロセッサ・アーキテクチャに対応する第２のプログラムを与えるステップと、

ｃ）前記第１及び第２のプロセッサ・アーキテクチャを示す複数の状態を有する制御転送モード手段を与えるステップと、

ｄ）次に実行される命令を有する前記プログラムに対応するプロセッサ・アーキテクチャを示す前記複数の状態の１つに、前記制御転送モード手段を設定するステップと、

ｅ）制御転送データをメイン・メモリまたはハードウェア・レジスタのどちらに送信するかを決めるために、前記制御転送モード手段の状態を使用するステップと、

ｆ）前記制御転送モード手段の状態に従って、前記第１及び第２のプログラム間で制御転送データを送信するス

テップと、を有する方法。
（12）前記制御転送モード手段は、情報の単一ビットの値にもとづいて前記第1及び第2のプロセッサ・アーキテクチャを示す、前記（11）記載の方法。
（13）前記第1のプロセッサ・アーキテクチャのアドレス範囲が前記第2のプロセッサ・アーキテクチャのアドレス範囲より小さい、前記（11）記載の方法。
（14）マイクロプロセッサ、及び外部バスで前記マイクロプロセッサに接続されている外部メモリを有するマイクロプロセッサ・システムの第1及び第2のプロセッサ・アーキテクチャ間にてプログラム制御を転送する方法であって、前記マイクロプロセッサは、前記外部メモリ、レジスタ及び実行ユニット間で接続されるバス及びキャッシュ・ユニットを有し、前記マイクロプロセッサは、自身が操作する前記レジスタにプロセス・スタックを格納するように構成され、前記方法は、
ａ）制御転送モード手段にて、前記第2のプロセッサ・アーキテクチャから前記第1のプロセッサ・アーキテクチャへの前記プログラム制御の転送を示すステップと、
ｂ）保持手段において前記第2のプロセッサ・アーキテクチャに関するデータを保持するステップと、
ｃ）前記プログラム制御を前記第2のプロセッサ・アーキテクチャから前記第1のプロセッサ・アーキテクチャに転送するステップと、
ｄ）前記第2のプロセッサ・アーキテクチャに関するデータを、前記プログラム制御の転送の際に前記保持手段から前記第2のプロセッサ・アーキテクチャに戻すステップと、を含む方法。
（15）ｅ）前記制御転送モード手段を、前記外部バスを介して前記外部メモリに適用するステップと、を含む、前記（14）記載の方法。
（16）前記プログラム制御を前記第2のプロセッサ・アーキテクチャから前記第1のプロセッサ・アーキテクチャに転送するステップが、
ｃ1）プログラムのアーキテクチャに従って、前記制御転送モード手段の状態を設定するステップと、
ｃ2）制御転送データが外部メモリまたはハードウェア・レジスタのどちらに送信されるかを決めるために前記制御転送モード手段の状態を使用するステップと、
ｃ3）前記制御転送手段の前記状態に従って、プログラム間で制御転送データを送信するステップと、を含む、前記（14）記載の方法。
（17）前記制御転送モード手段が情報の1ビットで前記プログラム制御の転送を示す、前記（14）記載の方法。
（18）前記第1のプロセッサ・アーキテクチャのアドレス範囲が前記第2のプロセッサ・アーキテクチャのアドレス範囲より小さい、前記（14）記載の方法。
（19）前記プログラム制御転送手段が前記第1のプロセッサ・アーキテクチャから前記第2のプロセッサ・アーキテクチャへの前記プログラム制御転送を示す場合、前記保持手段に前記第1のプロセッサ・アーキテクチャの基底アドレスのオフセットを保持するステップと、前記プログラム制御転送手段が前記第2のプロセッサ・アーキテクチャから前記第1のプロセッサ・アーキテクチャへの前記プログラム制御転送を示す場合、前記保持手段に前記第2のプロセッサ・アーキテクチャのアドレス・ビットを保持するステップと、を更に含む、前記（14）記載の方法。
（20）マイクロプロセッサと外部メモリとを有するマイクロプロセッサ・システムであって、前記マイクロプロセッサは、前記外部メモリ、レジスタ及び実行ユニット間で接続されるバス及びキャッシュ・ユニットとを有し、前記マイクロプロセッサは、自身が操作する前記レジスタにプロセス・スタックを格納するように構成され、前記マイクロプロセッサ・システムは、
ａ）第1の連係方法に従って第1の連係データを格納するように構成される第1のマイクロプロセッサ・アーキテクチャと、
ｂ）第2の連係方法に従って第2の連係データを格納するように構成される第2のマイクロプロセッサ・アーキテクチャと、
ｃ）前記第1の連係データを前記第2の連係データに変換する手段と、を有する、マイクロプロセッサ・システム。
（21）前記第2の連係データを前記第1の連係データに変換する手段を更に含む、前記（20）記載のマイクロプロセッサ・システム。
（22）前記連係データが第1及び第2のマイクロプロセッサ・アーキテクチャの一方によってメモリに記憶されることが必要ない場合、前記第1及び第2のマイクロプロセッサ・アーキテクチャの他方からの前記連係データをレジスタに保持する手段を更に含む、前記（20）記載のマイクロプロセッサ・システム。
（23）前記第2のマイクロプロセッサ・アーキテクチャは前記第1の連係方法をエミュレートする方法を更に含む、前記（20）記載のマイクロプロセッサ・システム。
（24）プログラム制御を転送する手段を含み、前記転送する手段が制御転送モード・フィールドを含み、前記フィールドが、前記第1及び第2のプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す、前記（20）記載のマイクロプロセッサ・システム。
（25）前記制御転送モード・フィールドは、前記マイクロプロセッサにより外部バスを介して前記外部メモリに書込まれる、前記（24）記載のマイクロプロセッサ・システム。
（26）前記制御転送モード・フィールドが情報の1ビットを含み、前記1ビットが、どのアーキテクチャが前

記プログラム制御を有するかを示す、前記（２４）記載のマイクロプロセッサ・システム。

（２７）前記第２のマイクロプロセッサ・アーキテクチャのアドレス範囲が、前記第１のマイクロプロセッサ・アーキテクチャのアドレス範囲より大きい、前記（２０）記載のマイクロプロセッサ・システム。

（２８）前記第１のマイクロプロセッサ・アーキテクチャがＸ８６　ＣＩＳＣアーキテクチャである、前記（２０）記載のマイクロプロセッサ・システム。

（２９）前記第２のマイクロプロセッサ・アーキテクチャがPowerPC ＲＩＳＣアーキテクチャである、前記（２０）記載のマイクロプロセッサ・システム。

（３０）複数のマイクロプロセッサ及びメイン・メモリを有するマイクロプロセッサ・システムであって、前記マイクロプロセッサは、前記メイン・メモリ、レジスタ及び実行ユニット間で接続されるバス及びキャッシュ・ユニットを有し、自身が操作する前記レジスタにプロセス・スタックを格納するように構成され、前記マイクロプロセッサ・システムは、

ａ）第１のマイクロプロセッサ・アーキテクチャと、

ｂ）第２のマイクロプロセッサ・アーキテクチャと、

ｃ）前記第１及び第２のプロセッサ・アーキテクチャの、プログラム制御を解放する一方から獲得する他方に前記プログラム制御を転送する記憶手段と、を含み、前記記憶手段が、前記プログラム制御を解放するプロセッサ・アーキテクチャからの情報を有し、該情報が、前記プログラム制御を獲得するプロセッサ・アーキテクチャによりアクセスされ得る、前記マイクロプロセッサ・システム。

（３１）前記記憶手段は、前記プログラム制御を獲得する前記プロセッサ・アーキテクチャによって前記情報がメモリに記憶されることを必要としない場合、少くとも１つのレジスタに前記情報を記憶する、前記（３０）記載のマイクロプロセッサ・システム。

（３２）前記第２のアーキテクチャが前記第１のアーキテクチャをエミュレートする手段を更に有する、前記（３０）記載のマイクロプロセッサ・システム。

（３３）前記記憶手段は、前記第１及び第２のマイクロプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す制御転送モード・フィールドを有する、前記（３０）記載のマイクロプロセッサ・システム。

（３４）前記メイン・メモリにある前記制御転送モード・フィールドが、前記複数のマイクロプロセッサの１つに転送される、前記（３３）記載のマイクロプロセッサ・システム。

（３５）前記制御転送モード・フィールドが情報の１ビットを含み、前記１ビットが、どのアーキテクチャがプログラム制御を有するかを示す、前記（３３）記載のマイクロプロセッサ・システム。

（３６）前記第２のマイクロプロセッサ・アーキテクチャのアドレス範囲が、前記第１のマイクロプロセッサ・アーキテクチャのアドレス範囲より大きい、前記（３０）記載のマイクロプロセッサ・システム。

（３７）前記第１のマイクロプロセッサ・アーキテクチャがＸ８６　ＣＩＳＣアーキテクチャである、前記（３０）記載のマイクロプロセッサ・システム。

（３８）前記第２のマイクロプロセッサ・アーキテクチャがPowerPC ＲＩＳＣアーキテクチャである、前記（３０）記載のマイクロプロセッサ・システム。

【図面の簡単な説明】

【図１】本発明のコンピュータ・システムを示すブロック図である。

【図２】本発明のバスに接続された異なるアーキテクチャのＣＰＵを示す図である。

【図３】従来技術のシステム概念図である。

【図４】本発明のシステム概念図である。

【図５】本発明のプロセス・スタックを示す図である。

【図６】本発明のプロセス・スタックを示す図である。

【図７】本発明のプロセス・スタックを示す図である。

【図８】本発明の再指定されたプロセス・スタックを説明する図である。

【図９】本発明の識別情報を示す図である。

【図１０】本発明の識別情報を示す図である。

【図１１】本発明の識別情報を示す図である。

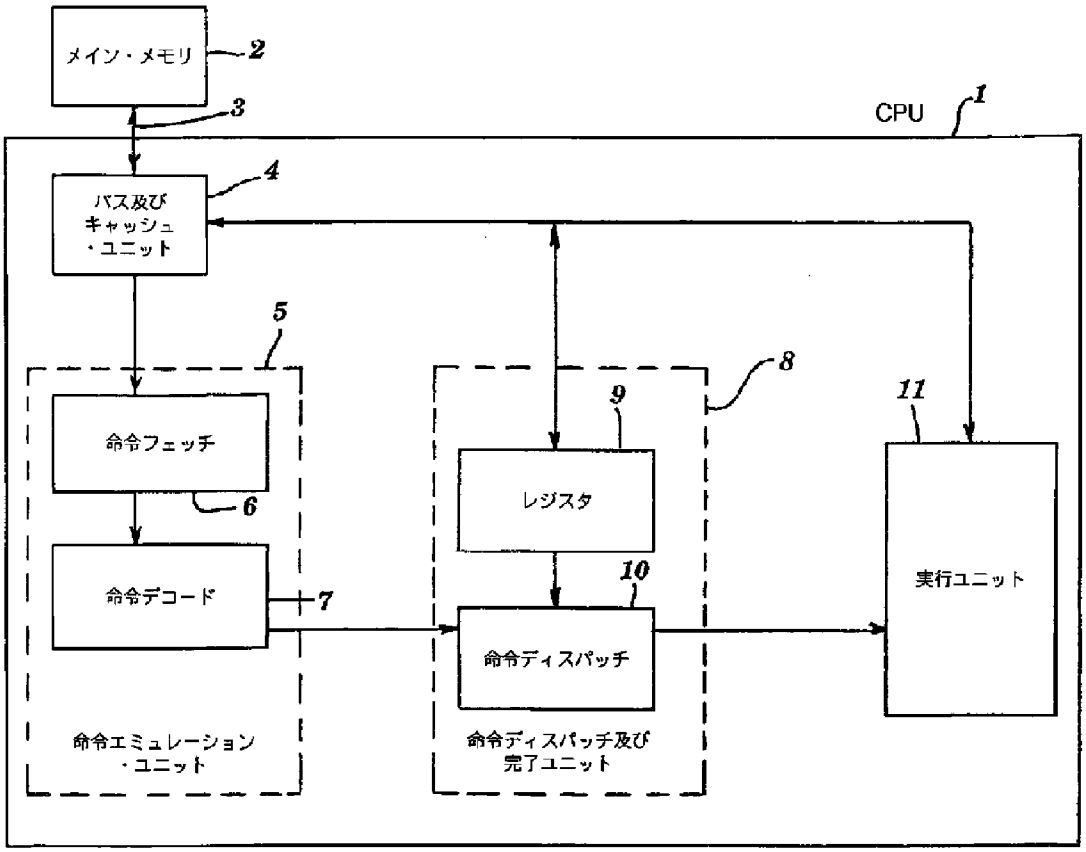
【図１２】本発明のレジスタを示す図である。

【図１３】スタックの位置とレジスタの関係を示す図である。

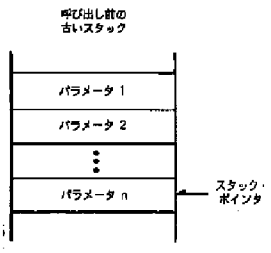
【符号の説明】

１　ＣＰＵ
２　メイン・メモリ
３　バス
４　バス及びキャッシュ・ユニット
５　命令エミュレーション・ユニット
６　命令フェッチ・ブロック
７　命令デコード・ブロック
８　命令ディスパッチ及び完了ユニット
９　レジスタ
１０　命令ディスパッチ・ブロック
１１　実行ユニット

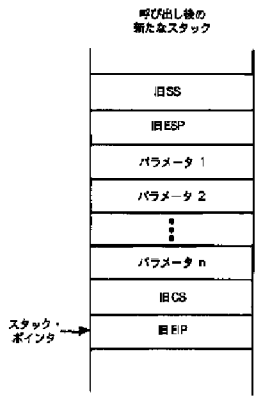
【図1】


メイン・メモリ
2
3
CPU
1
バス及び
キャッシュ
・ユニット
4
5
命令フェッチ
6
命令デコード
7
命令エミュレーション
・ユニット
8
9
レジスタ
10
命令ディスパッチ
命令ディスパッチ及び
完了ユニット
11
実行ユニット

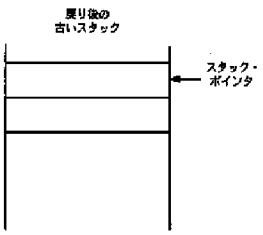
【図5】


呼び出し前の
古いスタック
パラメータ 1
パラメータ 2
パラメータ n
スタック・
ポインタ

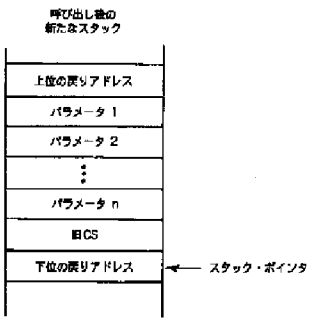
【図6】


呼び出し後の
新たなスタック
旧SS
旧ESP
パラメータ 1
パラメータ 2
パラメータ n
旧CS
旧EIP
スタック・
ポインタ

【図7】


戻り後の
古いスタック
スタック・
ポインタ

【図8】


呼び出し後の
新たなスタック
上位の戻りアドレス
パラメータ 1
パラメータ 2
パラメータ n
旧CS
下位の戻りアドレス
スタック・ポインタ

【図2】

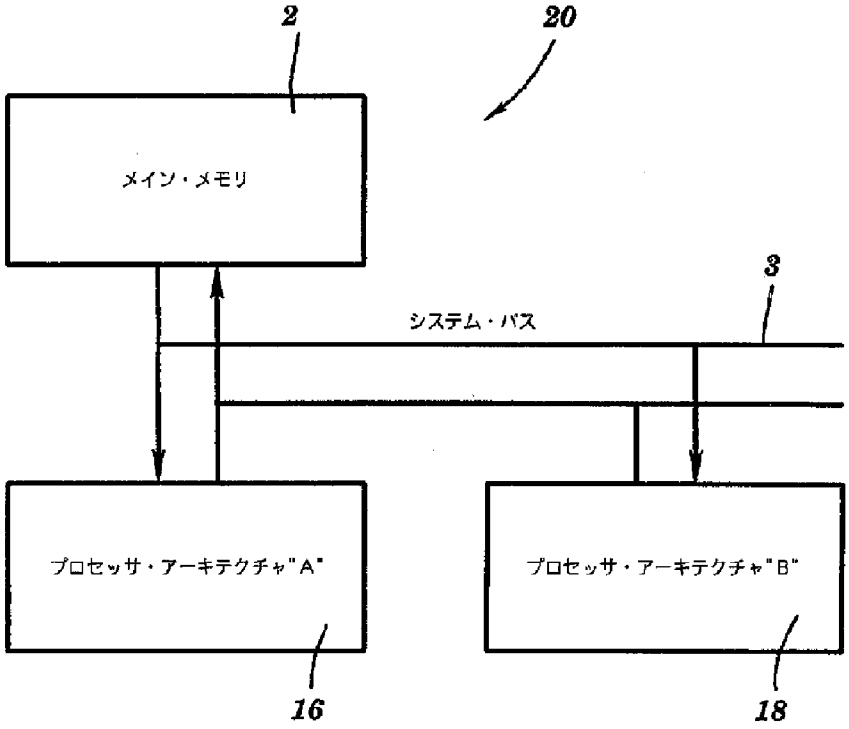
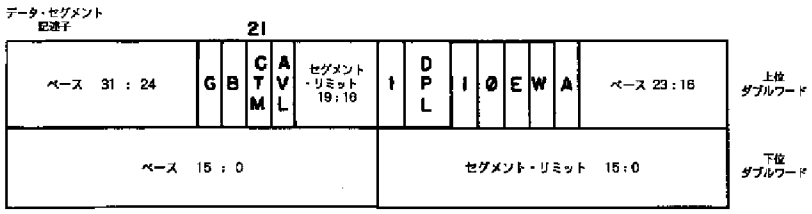
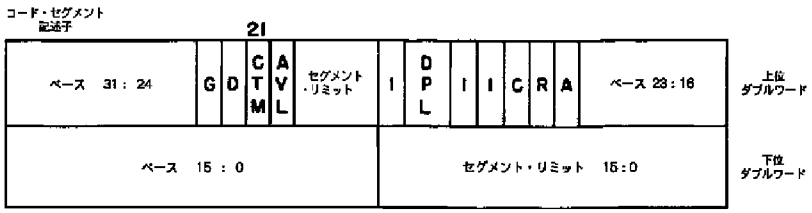

2
20
メイン・メモリ
3
システム・バス
プロセッサ・アーキテクチャ"A"
プロセッサ・アーキテクチャ"B"
16
18

図2

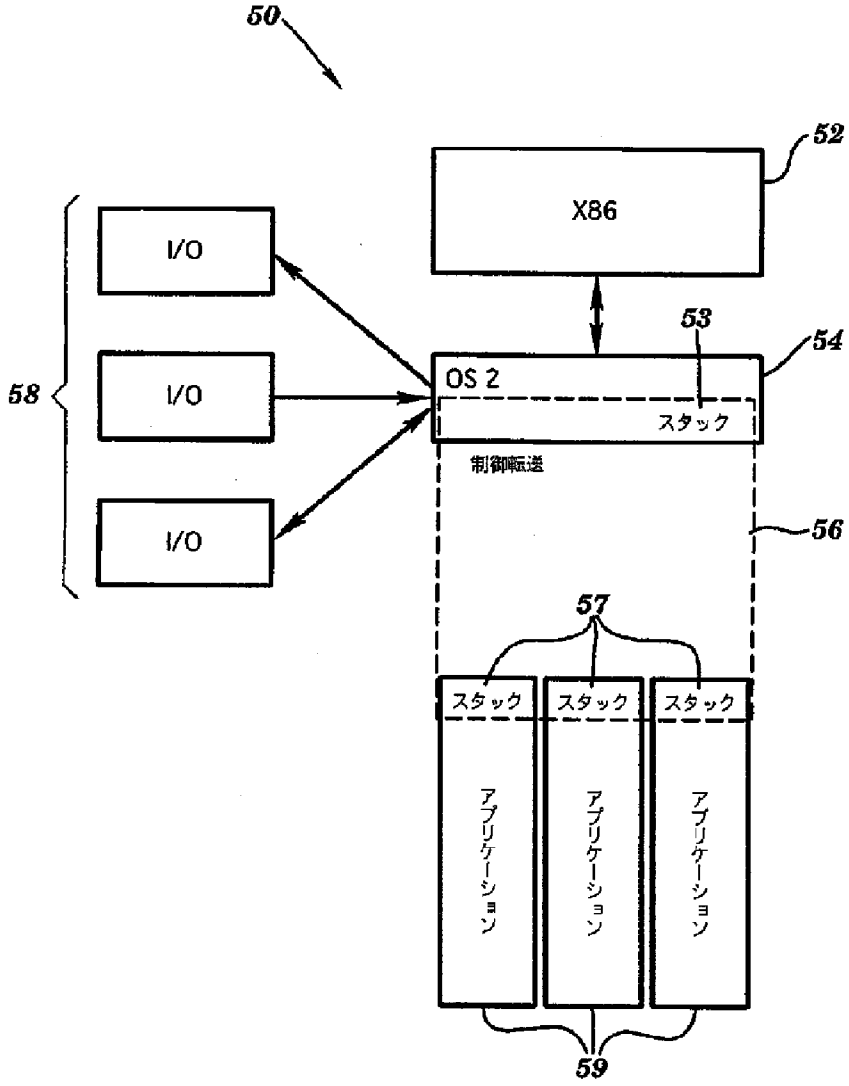
【図9】


データ・セグメント記述子
21
ベース 31：24
G
B
CTM
AVL
セグメント・リミット 19：16
P
DPL
I
Ø
E
W
A
ベース 23：16
上位ダブルワード
ベース 15：0
セグメント・リミット 15：0
下位ダブルワード

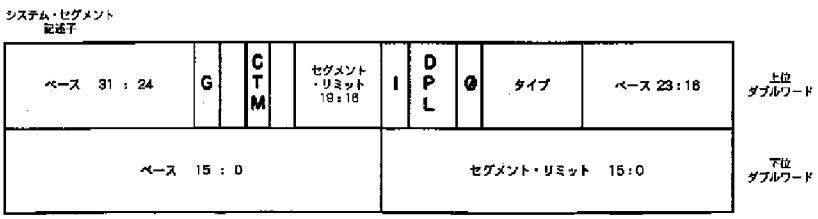
【図10】


コード・セグメント記述子
21
ベース 31：24
G
D
CTM
AVL
セグメント・リミット
P
DPL
I
I
C
R
A
ベース 23：16
上位ダブルワード
ベース 15：0
セグメント・リミット 15：0
下位ダブルワード

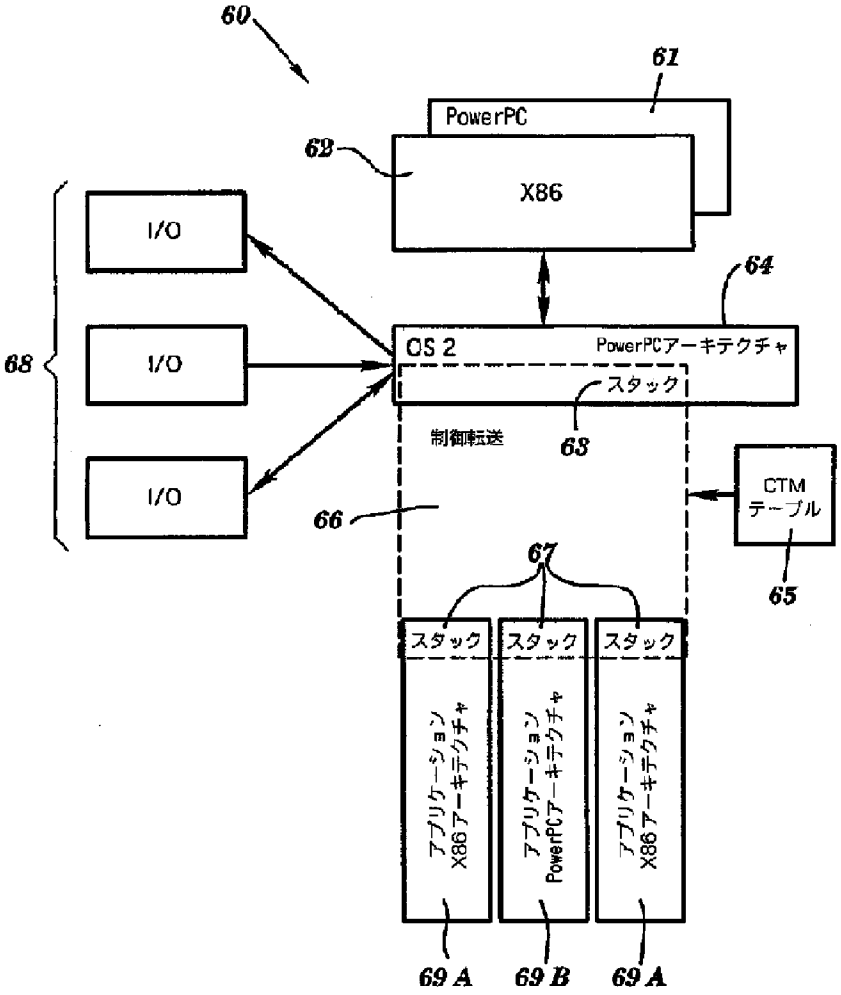
【図3】


50
52
X86
I/O
I/O
I/O
58
53
54
OS 2
スタック
制御転送
56
57
スタック
スタック
スタック
アプリケーション
アプリケーション
アプリケーション
59

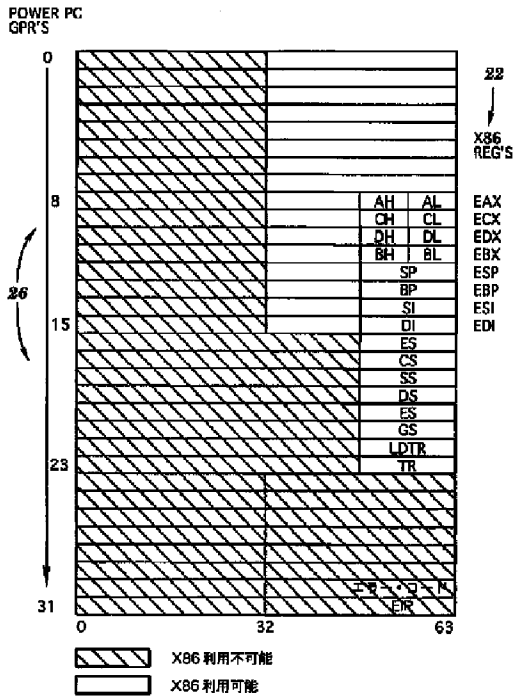
【図11】


システム・セグメント記述子
ベース 31 : 24
G
C T M
セグメント・リミット 19:16
I
D P L
0
タイプ
ベース 23:16
上位ダブルワード
ベース 15 : 0
セグメント・リミット 15:0
下位ダブルワード

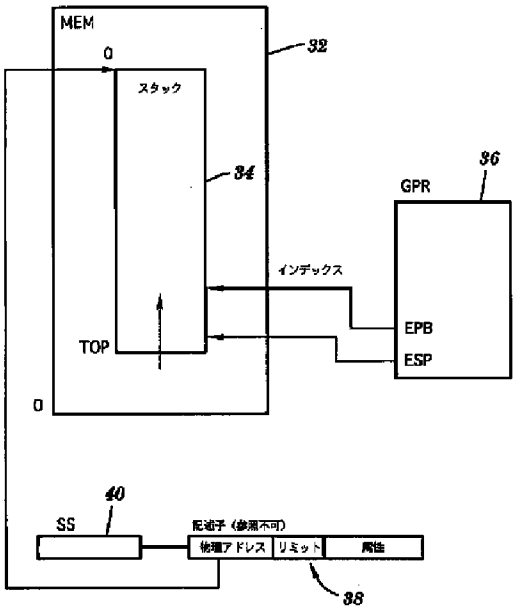
【図4】


60
61
PowerPC
62
X86
I/O
I/O
I/O
68
64
OS 2
PowerPCアーキテクチャ
スタック
63
制御転送
CTM
テーブル
65
66
67
スタック
スタック
スタック
アプリケーション
X86アーキテクチャ
アプリケーション
PowerPCアーキテクチャ
アプリケーション
X86アーキテクチャ
69A
69B
69A

【図12】



【図13】



フロントページの続き

(72)発明者　ジョン・エム・ケーティ
アメリカ合衆国05482、バーモント州シェルバーン、トレーシィ・アベニュー　21

(72)発明者　ステファン・ダブリュ・マーン
アメリカ合衆国05489、バーモント州アンダーヒル、スティーブンスビル・ロード、ボックス1395、レイル・ロード・ナンバー1